

LOSS IS ITS OWN REWARD: SELF-SUPERVISION FOR REINFORCEMENT LEARNING

Evan Shelhamer, Parsa Mahmoudieh, Max Argus, Trevor Darrell

UC Berkeley

{shelhamer,trevor}@cs.berkeley.edu; parsam@berkeley.edu; argus.max@gmail.com

ABSTRACT

Reinforcement learning, driven by reward, addresses tasks by optimizing policies for expected return. Need the supervision be so narrow? Reward is delayed and sparse for many tasks, so we argue that reward alone is a difficult and impoverished signal for end-to-end optimization. To augment reward, we consider a range of self-supervised tasks that incorporate states, actions, and successors to provide auxiliary losses. These losses offer ubiquitous and instantaneous supervision for representation learning even in the absence of reward. While current results show that learning from reward alone is feasible, pure reinforcement learning methods are constrained by computational and data efficiency issues that can be remedied by auxiliary losses. Self-supervised pre-training improves the data efficiency and policy returns of end-to-end reinforcement learning.

1 INTRODUCTION

End-to-end reinforcement learning (RL) addresses representation learning at the same time as policy optimization and value estimation. Of these dual pursuits, the focus has been on the reinforcement learning aspects of the problem such as stochastic optimization, exploration, and so forth. Having defined a loss on reward, the representation is delegated to backpropagation without further attention. However, representation learning is a bottleneck in current approaches bound by reward. Self-supervision acquaints the agent with the world beyond reward, so that self-supervised agents learn from all experience whether rewarded for it or not.

The swiftness of policy recovery from pre-training illustrates the critical role of representation. Retraining a decapitated agent, having destroyed the policy and value outputs while preserving the rest of the representation, is far faster than the initial training of the policy (Figure 1). Although the policy distribution and value function are lost, they are readily recovered given a representation from RL, even though the optimization and exploration issues remain. With the importance of representation established, we turn to self-supervision to take an ambient approach to RL attuned to reward and environment alike.

Self-supervision defines losses via surrogate annotations that are readily synthesized from bare, unlabeled inputs. In the context of RL, reward captures the task while self-supervision captures the environment. In this setting, every transition contributes gradients of ambient environmental signals. While loss from reward might be delayed and sparse, the losses from self-supervision are instantaneous and ubiquitous. Augmenting RL with these auxiliary losses enriches the representation through multi-task learning and improves policy optimization.

The purpose of self-supervision is representation learning and not modeling. While proxy tasks involve prediction, their purpose in our approach is to give gradients, and not necessarily furnish an effective generative model. We concentrate on auxiliary losses for state, dynamics, inverse dynamics, and reward that can be formulated in a discriminative fashion. We transfer pre-training by these self-supervised tasks to RL, and in the other direction we shed light on the contents of policy representations by examining transfer from RL to self-supervised tasks. Policy optimization to 90% of best return is sped-up 1.4 \times on average for a number of Atari environments.

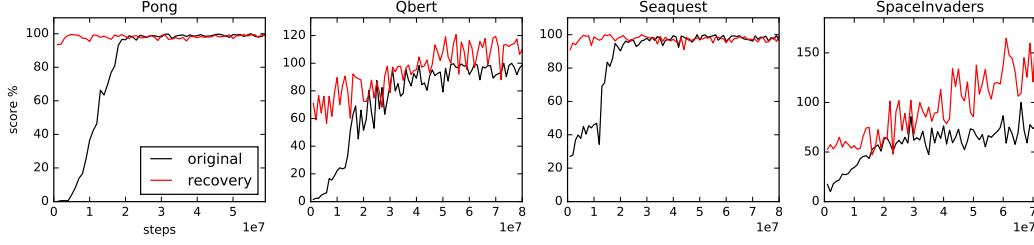


Figure 1: Current methods require many transitions to arrive at good policies, but policies are often quickly recovered from their representation. To separate reinforcement learning from representation learning, we decapitate the agent by destroying its policy and value output parameters, and then re-train end-to-end. Although the policy distribution and value estimates are obliterated, most of the parameters are preserved and the policy is swiftly recovered. The gap between the initial optimization and recovery illustrates a representation learning bottleneck.

2 PRELIMINARIES

Here we introduce background and notation.

2.1 REINFORCEMENT LEARNING

Reinforcement learning (RL) is concerned with policy optimization on Markov decision processes (MDPs).

Consider an MDP defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where \mathcal{S} is the set of states, \mathcal{A} is the set of actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the transition probability distribution, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the reward function, and $\gamma \in (0, 1)$ is the discount. In addition let p_0 be the distribution of the initial state s_0 .

Let π be a stochastic policy $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, and π_θ be a policy parameterized by θ .

The objective is to maximize the expected return of the policy $\eta(\pi)$:

$$\eta(\pi) = \mathbb{E}_{s_0, a_0, \dots} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) \right], \text{ where} \\ s_0 \sim p_0(s_0), a_t \sim \pi(a_t | s_t), s_{t+1} \sim P(s_{t+1} | s_t, a_t)$$

The expected return is represented by the state-action value function Q_π , the value function V_π , and the advantage function A_π :

$$Q_\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}, a_{t+1}, \dots} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) \right], \\ V_\pi(s_t) = \mathbb{E}_{a_t, s_{t+1}, \dots} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) \right], \\ \text{and } A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s), \text{ where } a_t \sim \pi(a_t | s_t), s_{t+1} \sim P(s_{t+1} | s_t, a_t)$$

Policy gradient methods iteratively optimize the policy return by estimating the gradient of return with respect to the policy parameters

$$g \triangleq \nabla_\theta \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r_t \nabla_\theta \log \pi_\theta(a_t | s_t) \right],$$

where the expectation is sampled by executing the policy in the environment. Instead of the discounted return itself, the policy gradient can make use of the advantage for actor-critic learning.

In this work we augment the policy gradient with auxiliary gradients from self-supervised tasks.

2.2 SELF-SUPERVISION

End-to-end RL admits policy learning in lieu of policy design in much the same way that end-to-end supervised learning has seen the advance of feature learning over feature design. Recently supervised learning, especially as carried out for computer vision, has seen the rise of deeper and higher-capacity networks trained by backpropagation, reaching networks of 100+ layers (He et al., 2016). These capacities are sustained only by massive amounts of annotation and other supervisory signals. Supervised pre-training on large-scale annotation as exemplified by ImageNet (Deng et al., 2009) currently delivers the most effective features for transfer learning to other tasks, but a wave of renewed interest in unsupervised and “self-supervised” learning offers alternatives that we catalogue here (Doersch et al., 2015; Noroozi & Favaro, 2016; Zhang et al., 2016; Donahue et al., 2016).

To illustrate the differences, consider the kinds of learning by their objectives

- supervised learning $\min_{\theta} \mathbb{E} [L_{\text{dis}}(f_{\theta}(x), y)]$
- unsupervised learning $\min_{\theta} \mathbb{E} [L_{\text{gen}}(f_{\theta}(x), x)]$
- self-supervised learning $\min_{\theta} \mathbb{E} [L_{\text{dis}}(f_{\theta}(x), s(x))]$ with surrogate annotation function $s(\cdot)$

for data x , annotation y , losses L either discriminative or generative, and parametric model f_{θ} .

Both unsupervised learning and self-supervision define losses without annotation, but unsupervised learning has historically focused on generative or reconstructive losses of the input alone, while nascent self-supervised methods instead define surrogate losses and synthesize the annotations from the data. (Note that this definition of self-supervision is strictly more general, since unsupervised learning is recovered by choosing the identity for the surrogate annotation.) Self-supervision is worth distinguishing from standard unsupervised learning by reconstruction not only for the change of perspective but for the effectiveness of the results. Witness improvements from visual self-supervision for transfer to many recognition tasks (Doersch et al., 2015; Noroozi & Favaro, 2016; Zhang et al., 2016; Donahue et al., 2016). These self-supervised methods can make use of more data for learning, but furthermore as auxiliary losses for RL they promise to make more use of the data already available to the policy.

3 SELF-SUPERVISION OF POLICIES

The state, action, successor, and reward (s, a, s', r) transition standard to RL admits many kinds of self-supervision. We explore the use of surrogate annotations that span different parts of the transitions to gauge what is informative for RL. These diverse, ambient signals mine further supervision from the same data available to existing RL methods.

Adopting self-supervision for RL raises issues of multi-task optimization and statistical dependence. Policy optimization and self-supervision need to be reconciled to learn from both reward and auxiliary losses without interference. Potential strategies include learning in stages, weighting the losses, or adapting policy trust region methods to the auxiliary gradients. In this work we take the simple approach of self-supervised pre-training followed by pure RL. As for the data distribution, in the RL setting the distribution of transitions is neither i.i.d. nor stationary, so self-supervision should follow the policy distribution. For pre-training we optimize auxiliary losses on the initial, random policy distribution, but further improvement by alternating or joint optimization is to be expected.

3.1 TASKS

For RL transfer, the self-supervised tasks must make use of the same transition data as RL while respecting architectural compatibility with the agent network. We first survey our auxiliary losses and then define their instantiations for our chosen environment and architecture. Every self-supervised task augments a common, agent-compatible encoder with a task-specific decoder. Once pre-training is complete the decoder is discarded and the shared representation is transferred to the initial agent network.

Refer to Figure 2 for diagrams of our tasks and architectures.

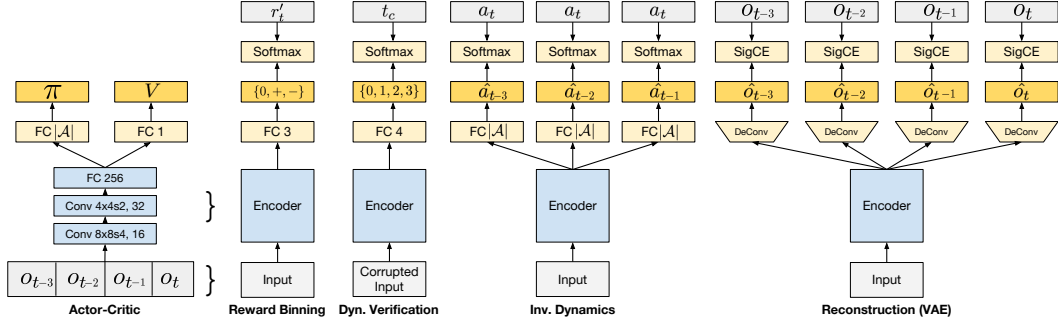


Figure 2: Architectures for reinforcement learning and self-supervision. The actor-critic architecture is based on A3C (Mnih et al., 2015) but with reduced capacity for experimental efficiency. The self-supervised architectures share the same encoder as the actor-critic for transferability. Each self-supervised task augments the architecture with its own decoder and loss.

Reward Self-supervision of reward is a natural choice to tune the representation for RL. Reward can be cast into a proxy task as instantaneous prediction by regression or binning into positive, zero, and negative classes. This is equivalent to one-step or zero-discount value function estimation, and so may seem redundant for value methods. However, the gradient of the instantaneous prediction task is less noisy because it is not subject to policy stochasticity or bootstrapping error. With reward, the proxy task accuracy is expected to closely mirror the degree of policy improvement. Our self-supervised reward task is to bin r_t into $r'_t \in \{0, +, -\}$ with equal balancing of the classes as done independently by Jaderberg et al. (2016).

Dynamics and Inverse Dynamics Even a single pair of transition suffices to define self-supervision for dynamics (successors) and inverse dynamics (actions). Surrogate annotations for such tasks capture state, action, and successor (s, a, s') relationships from transitions. These tasks need not form a transition model, and simple proxies can suffice to help tune the representation.

Dynamics can be cast into a verification task by recognizing whether state-successor (s, s') pairs are drawn from the environment or not (where negatives are synthesized by transplanting successors from other time steps). This can be made action contingent by extending the data to (s, a, s') and solving the same classification task. While the transition function is not necessarily one-to-one, and the negatives synthesized in this way are subject to noise, in expectation these surrogate annotations will match the transition statistics. Our self-supervised dynamics verification task is to identify the corrupted observation o_{t_c} in a history from t_0 to t_k , where o_{t_c} is corrupted by swapping it with $o_{t'}$ for $t' \notin \{t_0, \dots, t_k\}$.

Inverse dynamics, mapping $\mathcal{S} \times \mathcal{S} \rightarrow \mathcal{A}$, can be reduced to classification (for discrete actions) or regression (for continuous actions). When $|\mathcal{A}| \ll |\mathcal{S}|$, as is often the case, the self-supervision of inverse dynamics may be more statistically and computationally tractable. The difficulty of temporal self-supervision can be adjusted through the span and stride of time steps. Our self-supervised inverse dynamics task is to infer the intervening actions given a history of observations.

Reconstruction Auto-encoding (AE) and variational auto-encoding (VAE) learn to reconstruct the input subject to a representational bottleneck. The surrogate annotation for reconstruction is simply the identity as the loss is a distance between the input and output. While a popular line of attack for unsupervised learning, the representations learned by reconstruction are relatively poor for transfer (Donahue et al., 2016). Nevertheless we include reconstruction for comparison with our self-supervised tasks on distinct domains and ranges.

As illustrated by these proxy tasks, the auxiliary loss need not be generative, nor directly predictive of the transition and reward functions. Model-based RL is dedicated to learning transition and reward models, but we do not seek models of this kind nor improvements thereof. Furthermore the modeling in this work only serves the instrumental goal of representation learning. While full modeling could be intractable the gradients might suffice to improve reinforcement learning.

	Pong	Qbert	Seaquest	S. Invaders
Reward (F1)	0.73	0.66	0.03	0.38
Dyn. Ver. (acc.%)	97.5	92.8	95.0	90.5
... chance	25
Inv. Dyn. (acc. %)	34.2	17.5	25.5	33.3
... chance	16.6	16.6	5.5	16.6
VAE (ℓ_2)	2.34	2.31	3.10	2.04
... obs. mode	2.48	8.34	8.00	16.13

Table 1: We verify the feasibility of our collection of self-supervised tasks for Atari environments. All tasks reach reasonable performance on the chosen environments (with the exception of the VAE on Pong and reward on Seaquest). Task metrics improve through training and optimization converges quickly in less than ten epochs.

3.2 LOSS AS INTRINSIC REWARD

Intrinsic rewards are intended to scaffold skill learning, aid exploration, or otherwise guide the policy to improve (Barto et al., 2004; Chentanez et al., 2004). Rewards that operationalize novelty, curiosity, and competence focus on learning progress and predictive error (Schmidhuber, 1991; Oudeyer et al., 2005; Houthoofd et al., 2016). Self-supervisory losses could serve as intrinsic rewards of this kind, and by differentiability tune the representation at the same time.

The self-supervisory intrinsic reward could lead the policy distribution to novel and unsolved states for exploration. Learn the states that can be learned, until learned, then move on. Baseline is self-supervision on data distribution of fixed, uniform random policy.

Unifying loss and reward in this way is a new opportunity afforded by end-to-end RL.

4 RESULTS

We show results on self-supervision and policy pre-training for Atari. To begin we check the feasibility of the self-supervised tasks on transitions collected from random policies. Then for each proxy task and environment we measure improvements in return and data efficiency for self-supervised policy pre-training. As a probe into policy representations, we examine “reverse” transfer from reinforcement learning to proxy tasks with fixed weights. Policies pre-trained by self-supervision converge to same or better return and do so in fewer updates.

4.1 SELF-SUPERVISION

Our collection of self-supervised tasks are instantiated for Atari as variations of the Deep Q Network (DQN) architecture of Mnih et al. (2013). The standard DQN is taken as an encoder to which each task attaches its own decoder. To begin, the performance of the proxy tasks for their own sake is established for admissibility as pre-training for RL. Note however that proxy task performance need not be perfect to yield a transferable representation, and indeed proxy task accuracy has not been outstanding even for state-of-the-art self-supervised features Doersch et al. (2015).

In general the self-supervised tasks achieve reasonable performance and converge quickly. The task metrics across several ROMs are reported in Table 1.

Multi-task training of the reward, dynamics, and inverse dynamics tasks achieves comparable scores. It was not necessary to balance losses or otherwise tune learning for multi-task optimization of these auxiliary losses. Apparently the DQN-like encoder has enough capacity to jointly address these proxy tasks. A higher-capacity encoder may do better still, and the potential interaction of encoder capacity and policy optimization is worth investigating.

	Pong	Qbert	Seaquest	S. Invaders
Random Init.	21	16385	1750	1197
Calibrated Init.	95%	94%	100%	73%
Data-Dep. Init.	100%	95%	101%	75%
Reward	100%	109%	100%	191%
Dyn. Ver.	95%	122%	99%	111%
Inv. Dyn.	95%	112%	100%	95%
VAE	-	100%	100%	96%
Multi-task	100%	109%	97%	109%

Table 2: We compare returns achieved by self-supervised pre-training. Returns are reported as the absolute return for the baseline (random init.) and the return relative to the baseline for the other conditions. The returns achieved are nearly equal or better than baseline, with the strongest improvement given by reward self-supervision on Space Invaders.

	Pong	Qbert	Seaquest	S. Invaders
Data-Dep. Init.	1.18×	0.79×	1.03×	0.65×
Reward	1.46×	1.03×	1.02×	1.54×
Dyn. Ver.	1.51×	1.04×	1.03×	1.04×
Inv. Dyn.	0.44×	1.06×	1.07×	1.02×
VAE	0.01×	0.68×	0.97×	1.01×
Multi-task	1.49×	1.17×	1.01×	1.07×

Table 3: We examine the data efficiency of RL with self-supervised pre-training. To measure data efficiency, we calculate the area under the score/iteration curve and report the ratio to the baseline. Multi-task self-supervision improves 1.2× on average for these Atari environments. Focusing on early optimization, multi-task self-supervision gives 3× improvement for the first 10M iterations.

4.2 POLICY PRE-TRAINING

Self-supervised pre-training followed by RL fine-tuning is the simplest approach to incorporating auxiliary losses. This simplicity controls for confounds in joint optimization such as loss weighting and learning rate schedules. Any effect of self-supervision is purely due to representation learning prior to RL.

We compare simple initialization strategies—random initialization as well as calibrated and data-dependent initialization (Krähenbühl et al., 2016)—with our self-supervised tasks. These tasks include auxiliary losses that are agnostic to reward, letting learning make progress while waiting for reward. Perhaps surprisingly self-supervision of reward is not universally the most effective pre-training. Table 2 reports policy returns, Table 3 reports data efficiency, and Figure 3 shows policy optimization progress with the various pre-training schemes.

The immediate observation is that pre-training suffices to improve optimization. Returns at convergence for pre-training are nearly equal or better than baseline and data efficiency is improved. The sole exception is when pre-training diverges, but this is simple to diagnose. Pre-training is most helpful early in the optimization, when the policy distribution is close to the random distribution (which is the data distribution for pre-training).

There is no clear ordering of the tasks across environments, neither for policy return nor data efficiency. However, multi-task optimization of reward, dynamics, and inverse dynamics reliably delivers improvements on both fronts. Although more aggressive task tuning and search is possible, multi-task self-supervision is a practical default.

4.3 PROBING POLICY REPRESENTATIONS

Transfer from self-supervision to RL scaffolds the policy representation and improves optimization. However, whether transfer helps by capturing aspects of the environment or merely conditioning the

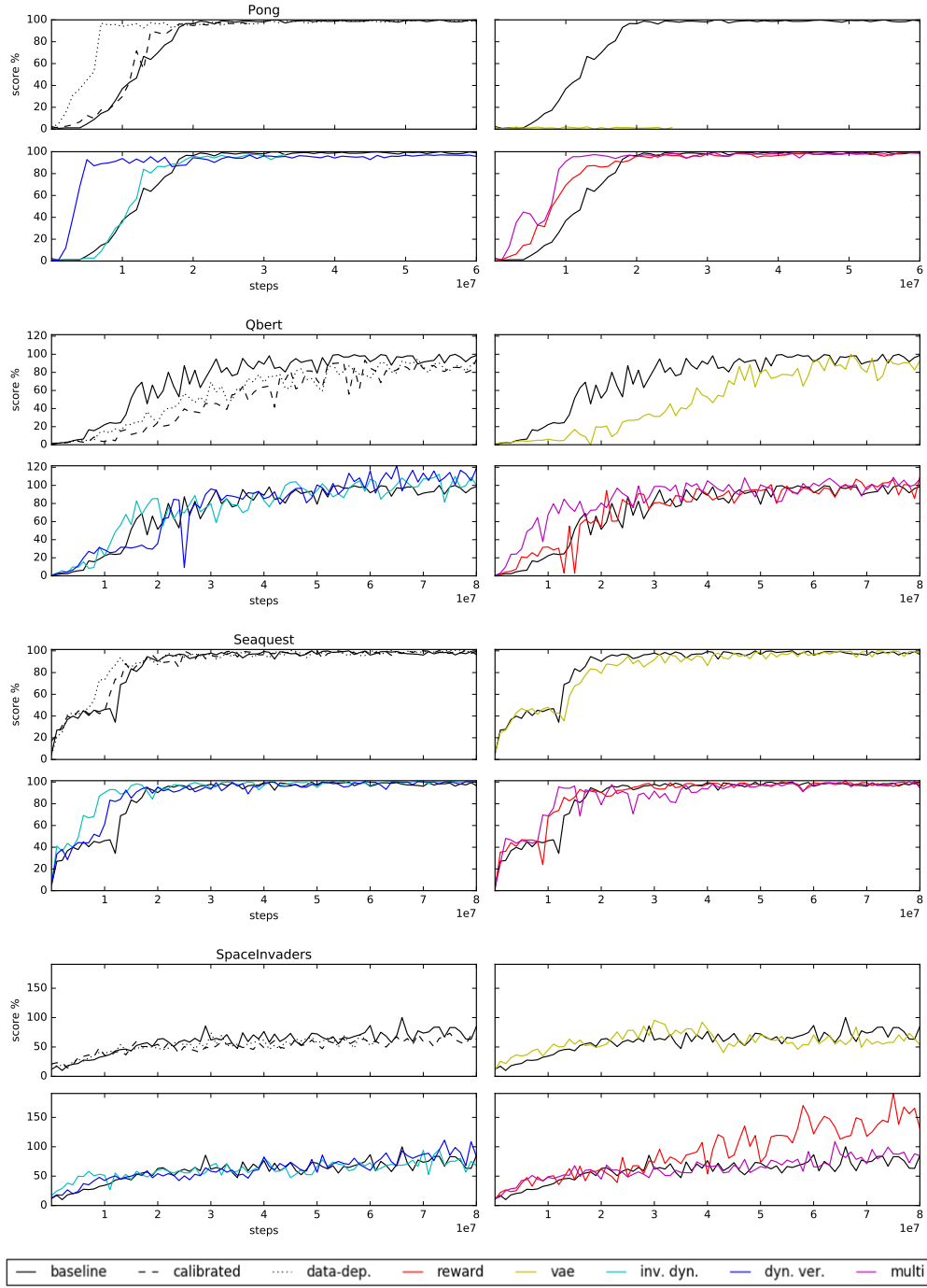


Figure 3: Optimization progress for reinforcement learning from self-supervised pre-training. Progress is reported as percentage of the best baseline return with evaluation every 1M updates. Many self-supervised tasks improve data efficiency without sacrificing return at convergence. Tasks independent of reward, such as dynamics and inverse dynamics, can nevertheless improve end-to-end RL. Improvement is strongest at the beginning of training when the pre-training and policy distributions are close. Refer to Section 3.1 for the details of the pre-training tasks.

weights is unclear. Furthermore, it is not obvious what is encoded by policy representations learned by pure RL. To gather indirect evidence, we explore transfer from RL to our proxy tasks to see

	Pong	Qbert	Seaquest	S. Invaders
Reward	$0.05\times$	$0.27\times$	-	$0.23\times$
VAE	-	$0.91\times$	$0.84\times$	$0.75\times$
Dyn. Ver.	$0.43\times$	$0.33\times$	$0.32\times$	$0.33\times$
Inv. Dyn.	$0.52\times$	$0.98\times$	$0.24\times$	$0.58\times$

Table 4: We analyze pure RL representations by learning decoders from the fixed RL representation to our self-supervised tasks. The relative quality of decoding from the fixed representation instead of end-to-end optimization gives some indication of what is captured by the RL features. Most proxy tasks suffer a significant ($> 50\%$) drop in accuracy, suggesting the representation is narrowly tuned to the RL task.

	Pong	Qbert	Seaquest	S. Invaders
Original DQN	20	1952	1705	581
Our A3C	21	16385	1750	1197

Table 5: We compare the best scores achieved by the original DQN (Mnih et al., 2013) and the same architecture optimized with our A3C implementation. Training is carried out for 80M updates. Scores are reported as the mean of 100 runs with random no-op starts as in existing work. This sanity check demonstrates reasonable returns, so improvement from self-supervision cannot be attributed to deficiencies in the RL setup.

which can be decoded from fixed parameters learned by RL. See Table 4 for the relative quality of the proxy tasks on the RL representation compared to end-to-end optimization.

Most proxy tasks suffer a significant ($> 50\%$) drop in accuracy. Although these same proxies can improve policy optimization, the policy representation itself is more specific to the RL task.

4.4 EXPERIMENTAL FRAMEWORK

Pre-training is carried out as straightforward supervised learning on fixed data shared by all of the tasks. The data for pre-training of each environment is collected by executing a random policy for 100,000 transitions. The transitions collected for self-supervision mirror the format of the transitions encountered during reinforcement learning. A number of episodes comprising roughly one-fifth of the transitions are held out for validation.

Transfer to reinforcement learning is carried out in the same manner across all tasks. First the self-supervised output layers are discarded and replaced by outputs for policy and value. The policy and value weights are initialized according to (LeCun et al., 1998) as in Mnih et al. (2015). To control for disparities in auxiliary losses, all networks are calibrated to equalize gradients across layers by the method of Krähenbühl et al. (2016). Without calibration transfer can fail to improve over random policy performance.

For architecture we follow the NIPS workshop edition of the DQN from Mnih et al. (2013) for its computational efficiency. For optimization we select the state-of-the-art asynchronous advantage actor-critic (A3C) method (Mnih et al., 2016) and configure it with comparable hyperparameters. For the environment we adhere to the specification from DeepMind (Mnih et al., 2015) by our own re-implementation through the OpenAI Gym (Brockman et al., 2016).

Table 5 checks our reinforcement learning baseline against the returns of the original DQN. Returns are better for all ROMs evaluated, justifying the baseline as reasonable for measuring further improvements due to self-supervision.

The code for the self-supervised tasks, policy optimization, and environment will be released.

5 RELATED WORK

Representation learning for reinforcement learning, robotics, and control is commonly known as state representation learning, as it yields the state for modeling the task as an MDP. That is, the goal of the state representation is to transform the history of observations, actions, and rewards into a sufficient state for efficient policy learning. This can be summarized formally as seeking a mapping ϕ such that the current state $s_t = \phi(o_{1:t}, a_{1:t}, r_{1:t})$ as in Jonschkowski & Brock (2015).

Unsupervised learning by auto-encoding is a common approach to state representation learning. The embed to control (Watter et al., 2015) objective combines variational auto-encoding and one-step dynamics modeling for image observations and locally-linear latent dynamics. The deep spatial auto-encoder (Finn et al., 2016) maps image observations into low-dimensional spatial coordinates by auto-encoding with a smoothness prior on the latent representation. The joint inverse and forward model of (Agrawal et al., 2016) learns to poke objects by self-supervising inverse dynamics while predicting future states (not observations) for regularization. These approaches optimize policies to achieve a goal state without a task reward, so it is not possible to fine-tune the representation to optimize return. In contrast we define simple to optimize auxiliary, discriminative losses that capture dynamics, inverse dynamics, and other aspects of the environment in tandem with RL.

For deep RL, the use of pre-training and transfer is limited. ML-DDPG (Munk et al., 2016) extends actor-critic with a one-step predictive model of the successor state and reward. The observation mapping is learned by the first layer of the model, transferred to the actor-critic network, and then fixed. Our successor self-supervision is discriminative rather than generative and we transfer all layers to the actor-critic network for end-to-end optimization. The end-to-end visuomotor policies of Levine et al. (2016) have the first layer initialized from supervised pre-training on ImageNet. Instrumented pose estimation pre-training further scaffolds the representation for policy optimization. Our auxiliary losses are purely self-supervised and only require regular transitions.

The robotic priors of Jonschkowski & Brock (2015) are auxiliary losses for temporal coherence, repeatability, proportionality, and causality. Multi-task optimization of these losses defines a linear, low-dimensional observation mapping for RL. These losses are defined on distances between states and conditioned on action and reward, whereas we define discriminative tasks on the (s, a, r, s') elements of transitions. A comparison of distance-based and discriminative auxiliary losses in the same setting could inform the choice of self-supervisory signals.

Concurrent work explores different methods to augment reinforcement learning with auxiliary losses. Jaderberg et al. (2016) extend value function estimation with instantaneous reward prediction and replay memory and introduce off-policy pseudo-reward control tasks. Mirowski et al. (2016) consider navigation tasks and auxiliary losses for spatial and path representations through coarse depth regression and a kind of loop closure for recognizing paths that have been already visited. Dosovitskiy & Koltun (2016) learn to predict future measurements of supervised, task-specific quantities such as the presence of enemies and health in a combat game (DOOM). In the same spirit as our work, these approaches seek to improve policy returns, data efficiency, and robustness of end-to-end RL. Our self-supervised tasks do not require additional privileged information, we focus on discriminative formulations of auxiliary losses, and we compare a variety of ambient signals for self-supervision.

6 DISCUSSION

It is encouraging that pre-training alone, with and without reward, can improve optimization for reinforcement learning. While pre-training mostly improves the early iterations, joint optimization is expected to further improve convergence.

Representation learning by self-supervision alone is agnostic to any particular task, and acts as a policy scaffold no matter the reward. This scaffold can be developed in the absence of an extrinsic reward whenever the policy is at play in the environment. Learning with self-supervisory losses improves data efficiency with respect to the extrinsic reward. A next step is to cast these losses into intrinsic rewards to further guide optimization. By augmenting RL with self-supervision, transitions without reward need not be so unrewarding for the representation.

ACKNOWLEDGEMENTS

This work was supported in part by Berkeley AI Research, Berkeley Deep Drive, NSF, DARPA, NVIDIA, and Intel. We gratefully acknowledge NVIDIA for GPU donation. We thank John Schulman and Chelsea Finn for advice and useful discussions.

REFERENCES

- Pulkit Agrawal, Ashvin Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *NIPS*, 2016.
- Andrew G Barto, Satinder Singh, and Nuttonpong Chentanez. Intrinsically motivated learning of hierarchical collections of skills. In *CDL*, pp. 112–119, 2004.
- Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.
- Nuttonpong Chentanez, Andrew G Barto, and Satinder P Singh. Intrinsically motivated reinforcement learning. In *NIPS*, pp. 1281–1288, 2004.
- J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.
- Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015.
- Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.
- Alexey Dosovitskiy and Vladlen Koltun. Learning to act by predicting the future. *arXiv preprint arXiv:1611.01779*, 2016.
- Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Deep spatial autoencoders for visuomotor learning. In *ICRA*, 2016.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- Rein Houthoofd, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Variational information maximizing exploration. In *NIPS*, 2016.
- Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.
- Rico Jonschkowski and Oliver Brock. Learning state representations with robotic priors. *Autonomous Robots*, 39(3):407–428, 2015.
- Philipp Krähenbühl, Carl Doersch, Jeff Donahue, and Trevor Darrell. Data-dependent initializations of convolutional neural networks. In *ICLR*, 2016.
- Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–48. Springer, 1998.
- Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.
- Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andy Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.
- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv*, 2013.

-
- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.
- Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *arXiv preprint arXiv:1602.01783*, 2016.
- Jelle Munk, Jens Kober, and Robert Babuška. Learning state representation for deep actor-critic control. 2016.
- Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.
- Pierre-Yves Oudeyer, Frédéric Kaplan, Verena V Hafner, and Andrew Whyte. The playground experiment: Task-independent development of a curious robot. In *AAAI Spring Symposium on Developmental Robotics*, pp. 42–47. Stanford, California, 2005.
- Jürgen Schmidhuber. Curious model-building control systems. In *IJCNN*, pp. 1458–1463. IEEE, 1991.
- Manuel Watter, Jost Springenberg, Joshka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *NIPS*, pp. 2746–2754, 2015.
- Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, 2016.